Schedulability Analysis for Multiresource Scheduling in Middleware for Distributed Real Time Systems

Radha Dongre
Sipna College of Engineering and Technology, Amravati

Anjali Mahajan
Government Polytechnic, Nagpur

ABSTRACT
Distributed Real Time Systems operate in resource-constrained environments and are composed of tasks that must process events and provide soft real-time performance. The main function of a computing system is to provide services to its users. In order to perform its function, a computing system uses various resources such as processors, memory, communication channels, etc. Managing and scheduling these resources is an important function.

General Terms
Middleware, Distributed real time system

Keywords
Schedulability analysis, real time system

1. INTRODUCTION
Real time systems are characterized by computational activities with timing constraints. Timing constraints in real time applications are predominantly soft in that deadlines may be missed as long as the long run fraction of the processing time allocated to each task in the application is in accordance with its utilization. A system design that can guarantee that deadline misses, if any, are bounded by constant amounts is sufficient to provide guarantees on long term processor shares. Hence, scheduling methods that ensure bounded deadline misses and that can be applied when other methods cannot are of considerable value and interest[1].

In a real-time system, the scheduling algorithm decides an order of execution of the tasks and also the quantity of time allowed to each task in the system so no task (for hard realtime systems), or a minimum number of tasks (for soft realtime systems), misses their deadlines. To verify if a scheduling policy guarantees the fulfillment of the temporal constraints of a task set, real-time systems designers use totally different exact or inexact schedulability conditions (also referred to as schedulability tests). The schedulability condition indicates if a given task set can be scheduled with a given scheduling algorithm such that no tasks within the set miss their deadlines. once a new task is created during a dynamic real-time system, an online admission management mechanism that uses a schedulability test, guarantees predictability if the new task is admitted. Examples of these kind of systems are those with Quality-of-Service (QoS) requirements, like multimedia system systems [1] [2], communication services [3][4], and automated control [5]. Different examples are found on the scheduling of real-time traffic over networks [6][1], or in open systems environments [7][8].

The schedulability test is based on the knowledge of the release times and the execution times of all tasks. The off-line schedulability test is useful when the system is highly deterministic, meaning that the release times and the execution times of all tasks are known, and either do not vary or vary only slightly. However, for systems in which tasks may arrive dynamically at run time, it is impossible to provide an off-line schedulability test. On-line admission controllers can perform the schedulability test at tasks' arrival times in dynamic systems. Two popular schedulability test approaches for end-to-end real-time tasks are time-demand analysis and schedulable utilization analysis. This research intends to propose a new approach for schedulability test for real-time scheduling for a single recourse, CPU. Second objective of this research work is to find out solution on the problem task scheduling in real time environment having heterogeneous resources[9].

Exact schedulability tests typically have time complexities and should not be adequate for on-line admission management if the system contains a great deal of tasks or a dynamic work. In contrast, most of the inexact schedulability tests provide low complexity sufficient schedulability tests that are suitable for use in on-line admission management mechanisms to decide the acceptance of the recently arriving tasks within the system[10]. If a task set doesn't satisfy a sufficient schedulability test, it's not known if the task set can be feasibly scheduled using a given scheduling policy. For this reason, it's important to determine that inexact schedulability test provides a far better performance, given the specific task set parameters.

In realtime operating systems, the scheduler is meant to produce optimal performance, optimum usage of resources, and fairness in resource assignment. In contrast, in real-time operating systems, the scheduler must restrict the non-determinism related to the concurrent system, and should offer the means to predict the worst-case temporal behaviour of the task set. A real-time scheduling algorithm provides an ordering policy for the execution of the tasks (as within the non-real-time programming algorithm)[11]. A given real-time scheduling algorithm may produce feasible or infeasible schedules. In an exceedingly possible schedule, each job for a given task set always completes by its deadline. In contrast, in an infeasible schedule, some jobs might miss a number of their deadlines. a group of jobs is schedulable in line with a given scheduling algorithm if, when using the algorithm, the scheduler always produces a possible schedule. The criterion used to measure the performance of the scheduling algorithms for real-time applications is their ability to find feasible schedules of the given application whenever such schedules exist. a hard real-time scheduling algorithm is optimal if, for any feasible task set, it always produces possible schedules [12].

The scheduling algorithms are often classified as static and dynamic. in an exceedingly static scheduling algorithm, all scheduling choices are provided a priori. For a given set of timing constraints, a table is constructed indicating the beginning and completion times of every task, such that, no task misses its deadline[13]. This approach is very

predictable, however once the parameters of the tasks change, the table should be recomputed and also the system restarted.

In dynamic scheduling algorithms, the scheduling decisions are taken at run-time based on the priorities of the tasks. These priority values are used to decide the execution order of the tasks. Priority values can be assigned statically or dynamically, depending on the dynamic scheduling algorithm.

The paper [12] introduced the first real-time scheduling algorithms for one processor (Rate-Monotonic and Earliest deadline First), and developed their corresponding schedulability analysis. RM assigns the highest priority to the task with the littlest amount, and EDF assigns priorities to the tasks considering the proximity of each instance of a task with its deadline, so that the task with the closest relative deadline receives the highest priority. Liu and Layland demonstrated that RM and EDF are optimal for fixed and dynamic priority algorithms, respectively.

2. GLOBAL SCHEDULAR

Moreover, global scheduling is particularly useful in case of open dynamic systems, where tasks may dynamically enter and leave the system. In fact, with static partitioning, every time a task enters the system, it must be allocated to a processor, and optimal allocation is an NP-Hard problem[14]. Therefore, admission control and allocation become difficult and time consuming. Also, when a task leaves the system, there may be the need for re-allocation and load balancing, and this reintroduces migration overhead. On the other hand, under global scheduling a task is not allocated to a processor. Therefore, when a task wants to enter the system, the only remaining problem is admission control, i.e. to understand if the task can be admitted into the system without jeopardizing the guarantee on the already admitted real-time tasks. This test is commonly referred to as schedulability test. In this paper we propose schedulability tests based on utilization and density bounds, which are polynomial in the number of tasks.
Using a technique similar to the one used in [15] for the EDF case, we then propose a schedulability test that, bounding the interference imposed on a task, is able to succesfully guarantee a larger portion of schedulable task sets, especially in presence of heavy tasks (i.e. tasks whose utilization is greater than 0.5). The global scheduling algorithm is given below

begin
for $\tau i = \tau 1$ to τm (m periodic task)
while there is a free processor Mf and an unassigned tasks do
    pick higher priority task
    assign (τi, Mf)
    if task executed within deadline return “success”
        else return “failure”
    endif
endwhile
Endfor

To achieve desired quality of service scheduling permits optimum allocation of resources among given tasks during a finite time. Formally, scheduling problem involves tasks that must be scheduled on resources subject to some constraints to optimize some objective function. The aim is to make a schedule that specifies when and on that resource every task are going to be executed [16]. It’s remained a subject of research in numerous fields for decades, may it be scheduling of processes or threads in an operating system, job shop, flow shop or company scheduling in production environment, printed circuit board assembly scheduling or scheduling of tasks in distributed computing systems like cluster, grid or cloud.
In recent years, distributed computing paradigm has gained abundant attention due to high scalability, reliability, and data sharing and low-cost than single processor machines. Cloud computing has emerged because the most popular distributed computing paradigm out of all others within the current situation. It provides on-demand access to shared pool of resources in a self-service, dynamically scalable and metered manner with guaranteed Quality of service to users. To produce guaranteed Quality of Service (QoS) to users, it's necessary that jobs should be efficiently mapped to given resources. If the required performance isn't achieved, the users will hesitate to pay.

2.1 Schedulability Test

A schedulability test defines a mathematical condition that's used to verify whether or not the task set meets its sequential restrictions for a given scheduling algorithm. The inputs of the test are the temporal parameters of the task set. A test is said to be sufficient within the sense that a task set is schedulable if it satisfies the test. However, if the task set doesn't satisfy the sufficient test, it's not known whether or not the task set are often schedulable using that scheduling algorithm. A test is claimed to be necessary if all schedulable task sets satisfy the test. Otherwise, if a given task set satisfies the test, we cannot say that it's schedulable. Actual tests offer a necessary and ample condition. The exact schedulability tests offer solely a sufficient (but not necessary) schedulability condition.
Schedulability tests depend on the scheduling algorithm chosen and the knowledge of the parameters of the task set. The schedulability test in dynamic scheduling algorithms is often performed off-line or on-line. If the test is executed off-line, there should be complete data of the set of tasks that are to be executed within the system along with the timing constraints imposed on each task (e.g., deadlines, precedence restrictions, execution times) before the execution of the system. During this case, the arrival of recent tasks isn't allowed whereas the system is executing, and therefore the tasks cannot change their timing constraints. In contrast, if the scheduling test is performed on-line, new arrivals are allowed at any time and therefore the tasks will change their timing constraints throughout the execution of the system. In this test, the scheduler decides dynamically, by means of an admission control mechanism, if the acceptance of those new tasks won't cause alternative tasks to miss their deadlines.
The utilization bound, for a given real time scheduling algorithm, is the value such that any task set, whose utilization factor is no larger than is schedulable under that scheduling algorithm. Utilization-based schedulability conditions verify if the utilization of the task set doesn't exceed the utilization bound.

3. COMPARISON OF SCHEDULABILITY ANALYSIS

We compare two methods for performing schedulability tests: Earliest deadline First(EDF) and Fixed Priority(FPT), when they are used in implemented admission control service.
Each of our experiments is characterized by a pair (m, n) where m is the number of processors and n is the task set size. We considered 40 different utilization levels {0.025m, 0.5m, . . . 0.975m,m} for each experiment (m, n). 15 experiments were conducted with m ∈ {2, 4, 6} and n ∈ {10, 20, 40, 60, 80} to compare the acceptance ratios of EDF and FPT tests. We present the acceptance ratios of the experiments with

parameters (m = 4, n = 20), (m = 4, n = 40), (m = 6, n = 20) and (m = 6, n = 80) in Figure 1 to Figure 4 , respectively.

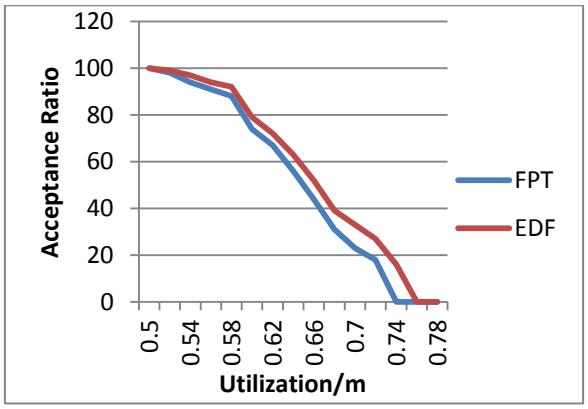

Figure 1 Acceptance Ratio of EDF and FPT tests for m=4, n=20

Each graph in Figure 1 to Figure 4 presents the acceptance ratio for both tests. The x-axis is the system utilization U/m for utilization level U and the y-axis represents the acceptance ratio. The acceptance ratios of both FPT and EDF tests are around 100% at relatively low utilization level (e.g., $U \leq 0.5m$) and 0% at very high utilization level (e.g., $U > 0.8m$). We plot the acceptance ratio in Figure 1 to Figure 4 for the utilization levels between 0.5m and 0.8m.

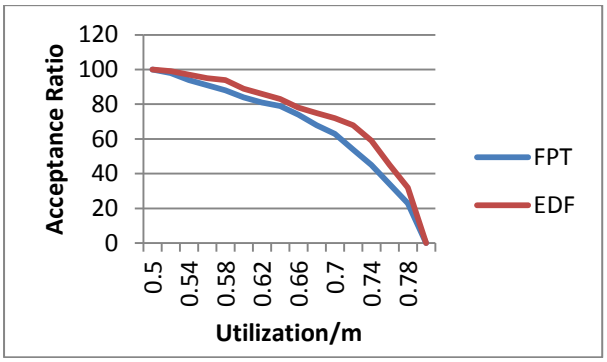

Figure 2: Acceptance Ratio of EDF and FPT tests for m=4, n=40

The acceptance ratios for both EDF test and FPT test are higher when the task set size increases for a given m. Notice that the acceptance ratio for both FPT and EDF tests in Figure 2 and Figure 4 are relatively "healthier" than that of in Figure 1 and Figure 3, respectively. A taskset with smaller cardinality having total utilization U has relatively higher number of high-utilization tasks in comparison to that of a taskset with larger cardinality having total utilization U. With higher number of high utilization tasks, the global FP scheduling suffers[14], and consequently, relatively smaller number of (low cardinality) tasksets passes both FPT and EDF tests in Figure 4 and Figure 6.

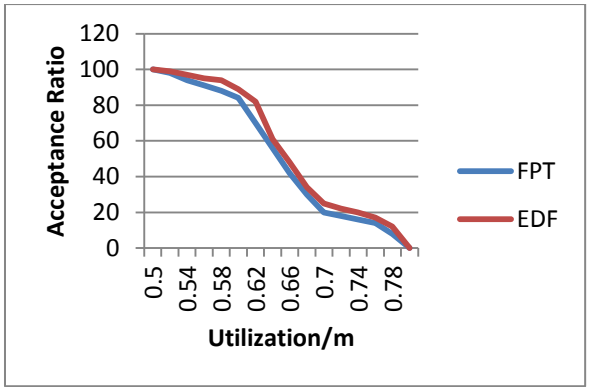

Figure 3 Acceptance Ratio of EDF and FPT tests for m=6, n=20

The improvement in acceptance ratio of the EDF test over the FPT test is noticeable at higher utilization level (e.g., $0.55m \leq U \leq 0.75m$) in all the four cases in Figure 1 to Figure 4. Both priority assignment policy and schedulability test play very important roles in determining the global FP schedulability of a task set at large utilization levels. The improvement in acceptance ratio of the EDF test at higher utilization levels is due to our improved priority assignment policy. For example, the acceptance ratio of the EDF test is around 30% higher than that of FPT test at utilization level 0.7m in Figure 4.

The EDF test outperforms the FPT test for tasksets with relatively larger cardinality for a given number of processors. The difference between the acceptance ratios of the EDF test and FPT test in Figure 2 and Figure 4 is considerably larger than that of in Figure 1 and Figure 3, respectively. The improvement of the EDF test over the FPT test increases when both task set cardinality and number of processors increases. The difference between the acceptance ratios of the EDF test and the FPT test at higher utilization level in Figure 4 is considerably larger than that of in Figure 1. When the number of tasks in a task set at a particular utilization level is relatively larger, there are relatively fewer tasks with large density. And, separating tasks based on the "highest-density" criterion of the FPT test is not that effective. Our proposed separation criterion for EDF test reduces the pessimism of interference on a lower priority tasks to a larger extent in comparison to the criterion of separating "highest-density" tasks of the FPT test. Thus, with increasing number of task set size, the EDF test performs significantly better than the FPT test.

The task set size (i.e., parameter n) in addition to parameters m and U has significant impact on global scheduling. The EDF test outperforms the FPT test for large n (i.e., where there are fewer large-density tasks in a task set). However, the performance of the EDF test with relatively smaller n is also significant; for example, the acceptance ratio of the EDF test is around 5% to 8% higher than that of the FPT test between utilization level 0.55m and 0.7m in Figure 3. Any improvement in acceptance ratio of the EDF test implies relatively lower demand on total processing capacity which in turn could significantly cut the cost of mass production of actual systems with relatively fewer numbers of processors.

The total synthetic utilization of all subtasks on each processor was changed systematically in different experiments, and randomly generated 60 task sets for each total synthetic utilization per processor. Note that the total synthetic utilization is calculated assuming there is a current instance of each (aperiodic or periodic) task on a processor which is different from the instantaneous synthetic

utilization on the processor at run time. Each set of tasks includes 4 aperiodic tasks and 5 periodic tasks with similar characteristics to the workloads used in earlier experiments. All tasks were made equally critical in this set of experiments.

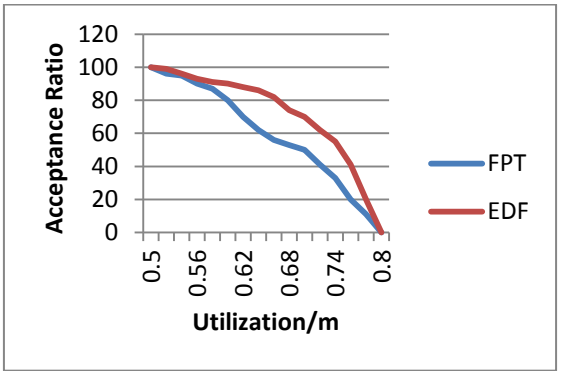

Figure 4 Acceptance Ratio of EDF and FPT tests for m=6, n=80

4. REFERENCES

[1] N. C. Audsley. 2001 On Priority Assignment in Fixed Priority Scheduling. Info. Proc. Letters, 79(1):39–44.

[2] T. P. Baker. 2006 An Analysis of Fixed-Priority Schedulability on a Multiprocessor. Real-Time Systems, 32(1-2):49–71.

[3] S. Baruah. 2007 Techniques for multiprocessor global schedulability analysis. In Proc of RTSS, pages 119–128.

[4] M. Bertogna and M. Cirinei. 2007 Response-Time Analysis for Globally Scheduled Symmetric Multiprocessor Platforms. In Proc. of RTSS, pages 149–160.

[5] M. Bertogna, M. Cirinei, and G. Lipari. 2009 Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms. IEEE Transactions on Parallel and Distributed Systems, 20(4):553–566.

[6] B. Brandenburg, J. Calandrino, and J. Anderson. 2008 On the Scalability of Real-Time Scheduling Algorithms on Multicore Platforms: A Case Study. Proc. of RTSS.

[7] R. Davis and A. Burns. 2011 A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. Accepted forpublication in the ACM Computing Surveys.

[8] J. C. Palencia and M. G. Harbour. 2008 Schedulability Analysis for Tasks with Static and Dynamic Offsets. In RTSS.

[9] S. Ramos-Thuel and J. P. Lehoczky. 2013 On-line scheduling of hard deadline aperiodic tasks in fixed-prioriry systems. In RTSS.

[10] B. Sprunt, L. Sha, and L. Lehoczky. 2009 Aperiodic task schedul-ing for hard real-time systems. The Journal of Real-Time Systems, 1(1):27–60.

[11] A. Srinivasan and J. Anderson. 2002 Optimal Rate-based Scheduling on Multiprocessors. In Proceedings of the 34th ACM Symposium on Theory of Computing.

[12] J. Strosnider, J. P. Lehoczky, and L. Sha. 1995 The deferrable server algorithm for enhanced aperiodic responsiveness in real-time environments. IEEE Transactions on Computers, 44(1):73–91.

[13] S. Wang, Y. Wang, and K. Lin. 2010 Integrating the Fixed Priority Scheduling and the Total Bandwidth Server for Aperiodic Tasks. In Proceedings of the International Conference on Real-Time Computing Systems and Applications.

[14] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen. 1999 A scalable solution to the multi-resource QoS problem. In Proc.of IEEE Real-Time Systems Symposium, Phoenix, AZ, USA.

[15] J. M. Lopez, J. L. Diaz, M. Garcia, and D. F. Garcia. 2000 Worst-case utilization bound for EDF scheduling on real-time multiprocessor systems. In Proc. 12th Euromicro Conf. Real-Time Systems, pages 25–33.

[16] A. Srinivasan and J. Anderson. 2002 Optimal rate-based scheduling on multiprocessors. In Proc. 34th ACM Symposium on Theory of Computing, pages 189–198. ACM.